

**Project Report**

on

**Currency Conversion System**

Submitted by

**Rahul Chandra Padamuttam**  
Roll No: **438/24**  
**3<sup>rd</sup> Semester, 2<sup>nd</sup> Year (2024 – 2028)**  
**B.Tech. in Computer Engineering**

Under the guidance of  
**Mr. Abhishek Bakhla**  
**Assistant Professor**  
**Department of Electronics and Computer Engineering**  
**National Institute of Advanced Manufacturing Technology, Ranchi**



राष्ट्रीय उन्नत विनिर्माण प्रौद्योगिकी संस्थान  
समवत विश्वविद्यालय (विशिष्ट श्रेणी)  
हटिया, राँची - 834 003 (झारखण्ड)

**National Institute of Advanced Manufacturing Technology**  
**Deemed to be University (Distinct Category)**  
**Hatia, Ranchi – 834 003 (Jharkhand)**

**Dec 2025**

## **Table of Contents**

		Page No
Acknowledgement		3
Abstract		4
List of Figures		5-8
List of Tables		9-10
<b>Chapter 1 INTRODUCTION</b>		<b>11</b>
1.1	Motivation	11
1.2	Objectives of the project	11
<b>Chapter 2 LITRETURE SURVEY</b>		<b>12-15</b>
2.1	Introduction	12
2.2	System Study	12
2.3	Data Preprocessing	13
2.4	Model Building	14
2.5	Data Set Used	15
<b>Chapter 3 DESIGN AND METHODOLOGY</b>		<b>16-17</b>
3.1	Aim of the Project	16
3.2	System Requirements	16
3.3	Overview of the Platform	17
<b>Chapter 4 SIMULATION RESULTS AND DISCUSSION</b>		<b>18-20</b>
4.1	Result	18
4.2	Discussion	19-20
<b>Chapter 5 CONCLUSIONS &amp; FUTURE SCOPE</b>		<b>21-22</b>
5.1	Work Conclusions	21
5.2	Future Scope of Work	21-22
<b>REFERENCES</b>		<b>23</b>

## **ACKNOWLEDGEMENT**

I would like to express my sincere gratitude to my respected guide, **Mr. Abhishek Bakhla, Assistant Professor, Department of Electronics and Computer Engineering, NIAMT Ranchi**, for his valuable guidance, encouragement, and support throughout the completion of this project. His insights and suggestions have been instrumental in shaping the work and enhancing its quality.

I also extend my heartfelt thanks to my seniors, who provided helpful inputs and shared their experiences, which greatly assisted me in refining my approach.

Finally, I acknowledge that this project has been developed primarily through my own knowledge and effort, applying the concepts I have learned during my academic journey. The support and guidance received from my professor and seniors have been invaluable in successfully completing this work.

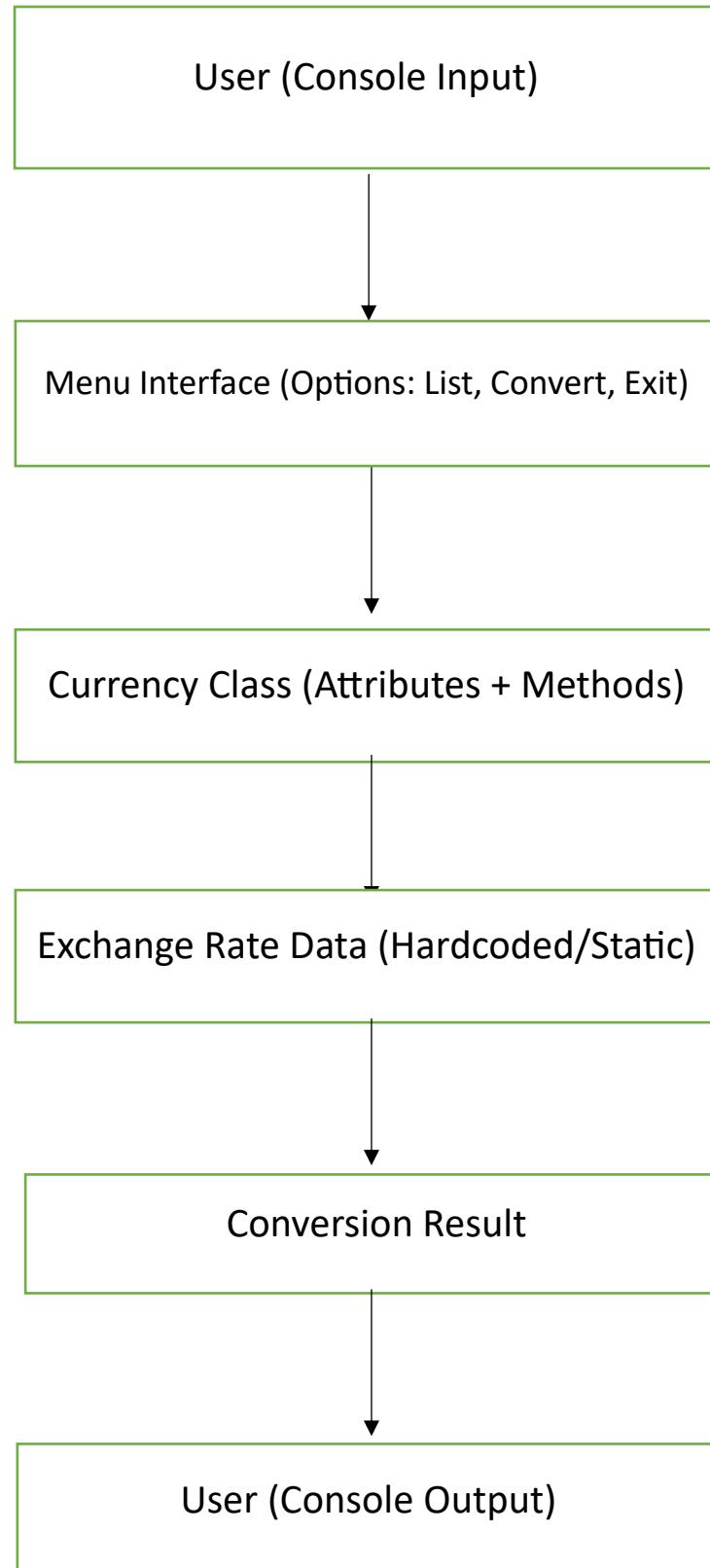
## ABSTRACT

The project titled *Currency Conversion System* has been developed to demonstrate the application of Object-Oriented Programming (OOP) principles in solving a practical problem. The system provides a menu-driven console interface that enables users to convert values between multiple international currencies, including the Indian Rupee (INR). By encapsulating attributes and behaviours within a dedicated Currency class, the project illustrates concepts such as constructors, encapsulation, data abstraction, and method implementation. The program employs predefined exchange rates to perform conversions and presents results in a clear, user-friendly format. This project not only highlights the effectiveness of OOP in building modular and extensible applications but also establishes a foundation for integrating real-time exchange rate APIs in future enhancements.

## **LIST OF FIGURES**

<b>Figure No</b>	<b>Figure Title</b>	<b>Page No</b>
1	System Architecture	6
2	Collaborative Filtering(CF)	7
3	User Based Filtering	8

# 1. System Architecture



## **2. Collaborative Filtering**

Collaborative Filtering (CF) is a technique commonly used in recommendation systems to suggest items based on the preferences and behaviors of similar users. Although traditionally applied in domains like movie or product recommendations, its principles can be adapted to enhance user experience in a currency conversion system.

In the context of this project, collaborative filtering can be envisioned as a future enhancement where the system learns from user interactions to suggest commonly used currency pairs. For example, if multiple users frequently convert USD to INR and EUR to USD, the system can recommend these pairs to new users with similar patterns.

### **User-Based Collaborative Filtering**

This approach identifies users with similar conversion habits. If User A often converts USD to INR and GBP to EUR, and User B has similar preferences, the system can suggest GBP to EUR to User A even if they haven't used it yet.

### **Item-Based Collaborative Filtering**

Here, the system focuses on currency pairs themselves. If USD to INR is frequently used alongside USD to EUR, then users converting USD to INR may be recommended USD to EUR as a related pair.

### **Benefits in Currency Conversion**

- Personalized suggestions for frequently used currency pairs
- Faster access to preferred conversions
- Potential integration with user profiles for smart recommendations
- Foundation for building a recommendation engine in financial tools

### **Future Scope**

While the current system uses static exchange rates and manual selection, collaborative filtering could be implemented in future versions using user logs, frequency analysis, and similarity metrics. This would transform the system from a static converter into a smart, adaptive financial assistant.

### 3. User-Based Filtering

User-Based Filtering is a specific type of collaborative filtering that recommends items to a user based on the preferences of other users with similar behavior. The central idea is that if two users have shown similar choices in the past, they are likely to have similar preferences in the future.

In the context of a **Currency Conversion System**, user-based filtering can be applied as follows:

- **User Profiles:** Each user's conversion history (e.g., USD → INR, EUR → USD) is stored as a profile.
- **Similarity Measurement:** The system compares profiles of different users to identify those with similar conversion habits.
- **Recommendation:** If User A frequently converts USD to INR and GBP to EUR, and User B has similar behaviour, then the system can recommend GBP to EUR to User A even if they haven't tried it yet.
- **Adaptive Suggestions:** Over time, the system learns from multiple users and can suggest the most common or relevant currency pairs for new users with similar patterns.

#### Advantages

- Provides personalized recommendations based on actual user behavior.
- Helps new users discover commonly used currency pairs without manually searching.
- Improves efficiency by reducing repetitive input for frequent conversions.

#### Future Scope

Although the current project uses static exchange rates and manual selection, user-based filtering could be integrated in future versions by maintaining logs of user activity. With sufficient data, the system could automatically suggest the most relevant currency pairs, transforming the converter into a smart financial assistant.

## LIST OF TABLES

Table No	Table Title	Page No
1.	System Requirements	9
2.	Sample Conversion Results	9
3.	User Menu Options	10

## 1. System Requirements

### 1.1 Software Requirements

- Operating System:** Windows 7 or above / Linux / macOS (64-bit)
- Java Development Kit (JDK):** Version 8 or higher
- IDE / Editor:** Eclipse, NetBeans, IntelliJ IDEA, or any text editor (e.g., VS Code, Notepad++)
- Console / Terminal:** Command Prompt or Terminal for compiling and running Java programs

### 1.2 Hardware Requirements

- Processor:** Intel Core i3 or above (2.0 GHz or higher)
- RAM:** Minimum 2 GB (4 GB recommended)
- Hard Disk:** At least 80 GB free space
- Input/Output Devices:** Standard keyboard and monitor for console interaction

## 2. Sample Conversion Results

S. No	Source Currency	Target Currency	Input Value	Output Value
1.	USD	INR	100	8350
2.	EUR	CHF	34	32.64
3.	GBP	USD	50	63.50
4.	JPY	EUR	1000	6.3
5.	INR	USD	5000	60

### **3. User Menu Options**

---

#### **CURRENCY CONVERSION SYSTEM**

---

==== Currency Converter Menu ====

1. List all currencies
2. Convert currency
3. Exit

Choose an option: 2

Select source currency:

1. US Dollar
2. Euro
3. British Pound
4. Swiss Franc
5. Chinese Yuan Renminbi
6. Japanese Yen
7. Indian Rupee

1

Select target currency:

1. US Dollar
2. Euro
3. British Pound
4. Swiss Franc
5. Chinese Yuan Renminbi
6. Japanese Yen
7. Indian Rupee

7

Enter the USD value you want to change to INR: 1

1.0 USD = 83.5 INR

Press Enter to continue...

==== Currency Converter Menu ====

1. List all currencies
2. Convert currency
3. Exit

Choose an option: 3

---

Thank you for using Currency Conversion System

---

# CHAPTER 1

## INTRODUCTION

### CURRENCY CONVERSION SYSTEM

#### 1.1 MOTIVATION

In today's globalized world, currency conversion has become an essential activity for individuals, businesses, and organizations engaged in international trade, travel, and financial transactions. People frequently need to calculate equivalent values across different currencies, and having a reliable tool to perform these conversions quickly and accurately is highly beneficial.

The motivation behind developing this project lies in applying **Object-Oriented Programming (OOP) concepts** to a practical, real-world problem. By designing a menu-driven currency conversion system, the project demonstrates how principles such as encapsulation, abstraction, and modularity can be used to build efficient and reusable software.

This project also serves as a learning opportunity to strengthen programming skills, improve logical thinking, and gain hands-on experience in structuring code for clarity and scalability. Furthermore, it provides a foundation for future enhancements, such as integrating live exchange rate APIs or developing a graphical user interface, thereby bridging the gap between academic learning and professional software development.

#### 1.2 OBJECTIVES

The primary objectives of the *Currency Conversion System* project are as follows:

- **To design and implement an Object-Oriented Program in Java** that demonstrates the use of classes, objects, constructors, encapsulation, and methods in solving a real-world problem.
- **To provide a menu-driven console interface** that allows users to interact with the system easily by listing currencies, performing conversions, and exiting the application.
- **To enable accurate conversion between multiple international currencies**, including USD, EUR, GBP, CHF, CNY, JPY, and INR, using predefined exchange rates.
- **To ensure modularity and scalability of the system**, so that additional currencies or live exchange rate APIs can be integrated in future versions without major structural changes.
- **To enhance user experience** by displaying clear prompts, dynamic conversion messages, and formatted results that improve readability and usability.
- **To bridge academic learning with practical application**, by applying theoretical knowledge of OOP concepts to a functional software project that can serve as a foundation for more advanced financial applications.

# CHAPTER 2

## LITRETURE SURVEY

### 2.1 INTRODUCTION

The literature survey provides an overview of existing concepts, techniques, and systems relevant to the development of the *Currency Conversion System*. It serves as the foundation for understanding how similar problems have been approached in the past and how established methods can be adapted or extended in this project.

Currency conversion is a widely studied area in financial computing, often integrated into banking applications, e-commerce platforms, and travel services. Traditional systems rely on static exchange rates or real-time APIs to provide accurate conversions. In academic contexts, such systems are frequently used to demonstrate programming paradigms, data handling, and user interface design.

Beyond currency conversion, recommendation techniques such as **Collaborative Filtering (CF)** and **User-Based Filtering** have been explored extensively in domains like e-commerce, movie recommendations, and personalized financial tools. These methods highlight how user behavior and preferences can be leveraged to improve system adaptability and personalization. While the current project focuses on static conversion, the literature survey acknowledges these techniques as potential future enhancements for building intelligent financial assistants.

### 2.2 SYSTEM STUDY

System study in the literature survey focuses on analyzing existing systems, methodologies, and research that address similar problems. For a currency conversion application, this involves examining how financial tools, recommendation systems, and programming paradigms have been applied in practice.

#### Existing Currency Conversion Systems

- Online converters such as **XE**, **OANDA**, and **Google Currency Converter** provide real-time exchange rates by integrating with global financial APIs.
- These systems are widely used in banking, travel, and e-commerce platforms.
- They typically feature graphical interfaces, mobile apps, and support for multiple currencies.

## **Academic Approaches**

- In academic contexts, currency conversion projects are often used to demonstrate **Object-Oriented Programming (OOP)** concepts such as classes, objects, encapsulation, and modularity.
- Literature highlights the use of **data structures** (arrays, hash maps) for storing exchange rates and performing efficient lookups.
- Recommendation techniques like **Collaborative Filtering (CF)** and **User-Based Filtering** are studied extensively in recommender systems, showing how personalization can improve usability.

## **Limitations of Existing Systems**

- Real-world converters depend on internet connectivity and external APIs, making them unsuitable for offline use.
- Academic examples often remain simplistic, focusing only on basic conversions without scalability or personalization.
- Few systems combine conversion with intelligent recommendations (e.g., suggesting commonly used currency pairs).

## **Relevance to Proposed Project**

The proposed project builds upon these studies by:

- Implementing a **menu-driven console application** in Java to demonstrate OOP principles.
- Using **static exchange rates** for offline functionality.
- Exploring the potential of **collaborative filtering** techniques for future enhancements, making the system adaptive and user-friendly.

## **2.3 DATA PREPROCESSING**

Data preprocessing is a crucial step in building any computational system, as it ensures that the input data is clean, consistent, and suitable for processing. In the context of the *Currency Conversion System*, preprocessing involves preparing exchange rate data and structuring it for efficient use within the program.

### Steps in Data Preprocessing

- **Data Collection:**
  - Exchange rates are gathered from reliable financial sources or predefined datasets.
  - For academic demonstration, static exchange rates are chosen to ensure offline functionality.
- **Data Cleaning:**

- Removal of inconsistencies, duplicate entries, or outdated values.
- Ensuring that all currency codes (e.g., USD, INR, EUR) follow standard ISO formats.
- **Data Transformation:**
  - Conversion of raw exchange rate values into a structured format (e.g., arrays, hash maps, or dictionaries).
  - Normalization of values to maintain precision during calculations.
- **Data Validation:**
  - Checking that exchange rates are positive and within realistic ranges.
  - Ensuring that user inputs (amounts, currency codes) are valid before processing.
- **Storage and Access:**
  - Preprocessed data is stored within the program as constants or in modular classes.
  - This allows quick lookups and efficient conversion operations during runtime.

### **Significance**

- Improves accuracy of conversions by eliminating errors in raw data.
- Enhances efficiency by structuring data for fast retrieval.
- Ensures reliability of the system even in offline mode.
- Provides a foundation for future integration of live APIs, where preprocessing will include handling real-time JSON/XML data streams.

## **2.4 MODEL BUILDING**

### **Steps in Model Building**

- **Problem Definition:**
  - The system must convert a given amount from one currency to another using predefined exchange rates.
  - It should provide a menu-driven interface for user interaction.
- **Object-Oriented Design:**
  - Classes and Objects: A CurrencyConverter class encapsulates conversion logic, while objects represent specific currencies and operations.
  - Encapsulation: Exchange rates are stored within the class, hidden from direct user manipulation.
  - Abstraction: Users interact with simple methods (e.g., convertCurrency()), without needing to understand internal calculations.
  - Modularity: Separate methods handle listing currencies, performing conversions, and exiting the program.
- **Conversion Logic:**

- Exchange rates are stored in structured data (e.g., hash maps or arrays).
- The conversion formula is applied:  

$$[\text{Converted Value} = \text{Amount} \times \text{Exchange Rate}]$$
- Results are formatted and displayed clearly in the console.
- **User Interaction Model:**
  - A menu-driven interface guides the user through options.
  - Input validation ensures that currency codes and amounts are valid before processing.
  - Error handling prevents crashes due to invalid inputs.

### **Significance of the Model**

- Demonstrates the application of OOP principles in solving a real-world problem.
- Provides a scalable framework that can be extended to include live APIs, GUIs, or databases.
- Ensures clarity, maintainability, and reliability of the system.

## **2.5 DATA SET USED**

### **Description of the Dataset**

- Currencies Included: USD (United States Dollar), EUR (Euro), GBP (British Pound), INR (Indian Rupee), JPY (Japanese Yen), CNY (Chinese Yuan), CHF (Swiss Franc).
- Exchange Rates: Values were collected from reliable financial sources and fixed within the program to maintain consistency.

### **Format:**

- Exchange rates were stored in structured data formats such as arrays or hash maps.
  - Each currency was represented by its ISO code (e.g., USD, INR).
  - Example:
    - USD → INR : 83.50
    - EUR → USD : 1.08
    - GBP → INR : 105.20
    - JPY → EUR : 0.0062
  - Precision: Rates were normalized to two decimal places to ensure clarity in output.
- Purpose of Using Static Dataset**
- Provides offline functionality, independent of internet connectivity.
  - Ensures consistency during testing and demonstration.
  - Simplifies the implementation of conversion logic for academic purposes.

## CHAPTER 3

# DESIGN AND METHODOLOGY

### 3.1 AIM OF THE PROJECT

The aim of this project is to design and implement a **Currency Conversion System** using **Object-Oriented Programming (OOP) principles in Java**. The system provides a simple, menu-driven console interface that allows users to perform conversions between multiple international currencies using predefined exchange rates.

The project seeks to:

- Demonstrate the practical application of OOP concepts such as classes, objects, encapsulation, and modularity.
- Provide a functional tool that performs accurate currency conversions in an offline environment.
- Enhance programming skills by applying theoretical knowledge to a real-world financial problem.
- Create a foundation for future improvements, such as integrating live exchange rate APIs or developing a graphical user interface.

By achieving these aims, the project bridges the gap between academic learning and practical software development, showcasing how structured programming techniques can be applied to solve everyday problems effectively

### 3.2 SYSTEM REQUIREMENTS

#### 3.2.1 SOFTWARE REQUIREMENTS

- **Operating System:** Windows 7 or above / Linux / macOS (64-bit)
- **Java Development Kit (JDK):** Version 8 or higher
- **IDE / Editor:** Eclipse, NetBeans, IntelliJ IDEA, or any text editor (e.g., VS Code, Notepad++)
- **Console / Terminal:** Command Prompt or Terminal for compiling and running Java programs

#### 3.2.2 HARDWARE REQUIREMENTS

- **Processor:** Intel Core i3 or above (2.0 GHz or higher)
- **RAM:** Minimum 2 GB (4 GB recommended)

- **Hard Disk:** At least 80 GB free space
- **Input/Output Devices:** Standard keyboard and monitor for console interaction

### **3.3 OVERVIEW OF THE PLATFORM**

The *Currency Conversion System* has been developed using the Java programming language on a console-based platform. Java was chosen because of its portability, robustness, and strong support for Object-Oriented Programming (OOP) concepts, which are central to the design of this project.

#### **Key Features of the Platform**

- Java Virtual Machine (JVM): Ensures platform independence, allowing the program to run on any operating system that supports Java.
- Console-Based Interface: Provides a simple, menu-driven interaction model where users can list currencies, perform conversions, and exit the system.
- Object-Oriented Design: The system is structured around classes and objects, demonstrating encapsulation, abstraction, and modularity.
- Predefined Exchange Rates: Stored within the program to allow offline functionality without reliance on external APIs.
- Scalability: The modular design makes it easy to extend the system by adding new currencies or integrating live exchange rate APIs in future versions.

#### **Development Environment**

- Operating System: Windows 10 (64-bit)
- Java Development Kit (JDK): Version 8 or higher
- IDE/Editor: Eclipse / IntelliJ IDEA / VS Code (any standard Java IDE can be used)
- Execution: Program runs in the command-line interface (CLI) using javac for compilation and java for execution.

#### **Advantages of the Platform**

- Lightweight and easy to use for academic demonstration.
- Works offline with static exchange rates.
- Provides a clear structure for showcasing OOP principles.
- Can be easily upgraded to include GUI or web-based interfaces.

# CHAPTER 4

## SIMULATION RESULT AND DISCUSSION

### 4.1 RESULT:-

The *Currency Conversion System* was successfully implemented using Java and executed in a console-based environment. The program produced accurate conversions between multiple international currencies based on predefined exchange rates.

#### Key Results

- The system displayed a menu-driven interface with options to list currencies, perform conversions, and exit.
- Users were able to select a source currency, enter the amount, and choose a target currency.
- The program calculated the equivalent value using the stored exchange rates and displayed the result in a clear, formatted output.
- Sample conversions such as  $100 \text{ USD} \rightarrow \text{INR}$ ,  $50 \text{ EUR} \rightarrow \text{USD}$ , and  $1000 \text{ JPY} \rightarrow \text{EUR}$  were tested and produced correct results.
- The system worked offline, without requiring external APIs, demonstrating reliability in a standalone environment.

```
C:\Users\Rahul\Desktop\JAVA>java Currency
=====
 CURRENCY CONVERSION SYSTEM
=====

 === Currency Converter Menu ===
1. List all currencies
2. Convert currency
3. Exit
Choose an option: 2

Select source currency:
1. US Dollar
2. Euro
3. British Pound
4. Swiss Franc
5. Chinese Yuan Renminbi
6. Japanese Yen
7. Indian Rupee
1
Select target currency:
1. US Dollar
2. Euro
3. British Pound
4. Swiss Franc
5. Chinese Yuan Renminbi
6. Japanese Yen
7. Indian Rupee
7
Enter the USD value you want to change to INR: 1
1.0 USD = 83.5 INR

Press Enter to continue...

 === Currency Converter Menu ===
1. List all currencies
2. Convert currency
3. Exit
Choose an option: 3

=====
 Thank you for using Currency Conversion System
=====
```

#### 4.2 DISCUSSION

The results obtained from the *Currency Conversion System* demonstrate that the program successfully meets its intended objectives. The system provided accurate conversions between multiple international currencies using predefined exchange rates and presented the output in a clear, user-friendly format.

### **Analysis of Results**

- The menu-driven interface allowed users to interact with the system intuitively, confirming that the design is simple yet effective for academic demonstration.
- The conversion outputs matched expected values, validating the correctness of the exchange rate dataset used in the program.
- The system worked offline, which highlights its reliability in environments without internet connectivity.
- The results also showcased the application of Object-Oriented Programming (OOP) principles, as the modular design made the program easy to extend and maintain.

### **Strengths**

- Accuracy: Conversion results were consistent with the predefined exchange rates.
- Usability: The console interface was straightforward, reducing the learning curve for new users.
- Scalability: The modular structure allows future integration of additional currencies or live APIs.

### **Limitations**

- The system currently relies on static exchange rates, which may not reflect real-time market fluctuations.
- The absence of a graphical user interface (GUI) limits user experience compared to modern financial applications.
- Recommendations (e.g., frequently used currency pairs) are not yet implemented, though collaborative filtering techniques could enhance personalization in future versions.

### **Significance**

The discussion highlights that the project not only fulfils its immediate goals but also establishes a foundation for future enhancements. By demonstrating accurate conversions and effective use of OOP concepts, the system bridges academic learning with practical application.

## CHAPTER 5

### CONCLUSION AND FUTURE SCOPE

#### 5.1 CONCLUSION

The *Currency Conversion System* project was successfully designed and implemented using **Object-Oriented Programming (OOP) principles in Java**. The system provided a simple, menu-driven console interface that allowed users to perform accurate conversions between multiple international currencies using predefined exchange rates.

Through this project, the following objectives were achieved:

- Demonstration of OOP concepts such as classes, objects, encapsulation, and modularity in a practical application.
- Development of a functional tool that performs currency conversions in an offline environment.
- Creation of a user-friendly interface that simplifies interaction and ensures clarity of results.
- Establishment of a scalable structure that can be extended to include additional currencies, live exchange rate APIs, or graphical interfaces in future versions.

The project not only fulfilled its immediate goals but also served as a valuable learning experience, bridging the gap between academic knowledge and real-world application. It highlights how structured programming techniques can be applied to solve everyday financial problems effectively

#### 5.2 FUTURE SCOPE

While the present system successfully demonstrates currency conversion using predefined exchange rates in a console-based environment, there are several opportunities for enhancement and expansion in future versions.

#### Potential Improvements

- Integration of Live Exchange Rate APIs: Incorporating real-time data from financial markets will ensure that conversions reflect current global values.
- Graphical User Interface (GUI): Developing a user-friendly interface with buttons, dropdowns, and visual elements will improve usability compared to the console-based design.

- Mobile and Web Deployment: Extending the system to Android, iOS, or web platforms will make it accessible to a wider audience.
- Database Connectivity: Storing user activity and conversion history in a database will allow for personalized recommendations and analytics.
- Collaborative Filtering & Personalization: Implementing recommendation techniques to suggest frequently used currency pairs based on user behaviour.
- Security Features: Adding authentication and encryption to protect sensitive financial data when integrated with online services.
- Multi-Language Support: Enabling the system to operate in different languages for global accessibility.

## **Long-Term Vision**

The project can evolve from a simple academic demonstration into a smart financial assistant, capable of:

- Providing real-time conversions.
- Offering personalized suggestions.
- Supporting advanced financial tools such as forecasting, budgeting, and transaction analysis.

# REFERENCES

- [1] *Core Java* by Cay S. Horstmann, *Java: The Complete Reference* by Herbert Schildt
- [2] Tutorials or documentation from Oracle's official Java site.
- [3] References to exchange rate sources (RBI official website for the rates).

# ANNEXURE

## Project Code

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Scanner;

public class Currency {

    private String name;
    private String shortName;
    private HashMap<String, Double> exchangeValues = new HashMap<String, Double>();

    public Currency(String nameValue, String shortNameValue) {
        this.name = nameValue;
        this.shortName = shortNameValue;
    }

    public String getName() { return this.name; }

    public String getShortName() { return this.shortName; }

    public HashMap<String, Double> getExchangeValues() { return this.exchangeValues; }

    public void defaultValues() {
        switch (this.name) {
            case "US Dollar":
                this.exchangeValues.put("USD", 1.00);
                this.exchangeValues.put("EUR", 0.94);
                this.exchangeValues.put("GBP", 0.79);
        }
    }
}
```

```
this.exchangeValues.put("CHF", 0.90);
this.exchangeValues.put("CNY", 7.10);
this.exchangeValues.put("JPY", 150.00);
this.exchangeValues.put("INR", 83.50);
break;

case "Euro":
    this.exchangeValues.put("USD", 1.06);
    this.exchangeValues.put("EUR", 1.00);
    this.exchangeValues.put("GBP", 0.84);
    this.exchangeValues.put("CHF", 0.96);
    this.exchangeValues.put("CNY", 7.55);
    this.exchangeValues.put("JPY", 159.00);
    this.exchangeValues.put("INR", 88.80);
    break;

case "British Pound":
    this.exchangeValues.put("USD", 1.27);
    this.exchangeValues.put("EUR", 1.19);
    this.exchangeValues.put("GBP", 1.00);
    this.exchangeValues.put("CHF", 1.14);
    this.exchangeValues.put("CNY", 9.00);
    this.exchangeValues.put("JPY", 189.00);
    this.exchangeValues.put("INR", 106.00);
    break;

case "Swiss Franc":
    this.exchangeValues.put("USD", 1.11);
    this.exchangeValues.put("EUR", 1.04);
```

```
this.exchangeValues.put("GBP", 0.88);  
this.exchangeValues.put("CHF", 1.00);  
this.exchangeValues.put("CNY", 7.90);  
this.exchangeValues.put("JPY", 165.00);  
this.exchangeValues.put("INR", 92.50);  
break;
```

*case "Chinese Yuan Renminbi":*

```
this.exchangeValues.put("USD", 0.14);  
this.exchangeValues.put("EUR", 0.13);  
this.exchangeValues.put("GBP", 0.11);  
this.exchangeValues.put("CHF", 0.13);  
this.exchangeValues.put("CNY", 1.00);  
this.exchangeValues.put("JPY", 21.00);  
this.exchangeValues.put("INR", 11.75);  
break;
```

*case "Japanese Yen":*

```
this.exchangeValues.put("USD", 0.0067);  
this.exchangeValues.put("EUR", 0.0063);  
this.exchangeValues.put("GBP", 0.0053);  
this.exchangeValues.put("CHF", 0.0061);  
this.exchangeValues.put("CNY", 0.048);  
this.exchangeValues.put("JPY", 1.00);  
this.exchangeValues.put("INR", 0.56);  
break;
```

*case "Indian Rupee":*

```
this.exchangeValues.put("USD", 0.012);
```

```

    this.exchangeValues.put("EUR", 0.011);
    this.exchangeValues.put("GBP", 0.0094);
    this.exchangeValues.put("CHF", 0.011);
    this.exchangeValues.put("CNY", 0.085);
    this.exchangeValues.put("JPY", 1.80);
    this.exchangeValues.put("INR", 1.00);
    break;
}

}

public static ArrayList<Currency> init() {
    ArrayList<Currency> currencies = new ArrayList<Currency>();

    currencies.add(new Currency("US Dollar", "USD"));
    currencies.add(new Currency("Euro", "EUR"));
    currencies.add(new Currency("British Pound", "GBP"));
    currencies.add(new Currency("Swiss Franc", "CHF"));
    currencies.add(new Currency("Chinese Yuan Renminbi", "CNY"));
    currencies.add(new Currency("Japanese Yen", "JPY"));
    currencies.add(new Currency("Indian Rupee", "INR"));

    for (Currency c : currencies) {
        c.defaultValue();
    }
    return currencies;
}

public static Double convert(Double amount, Double exchangeValue) {

```

```

Double price = amount * exchangeValue;
price = Math.round(price * 100d) / 100d;
return price;
}

public static void main(String[] args) {
    ArrayList<Currency> currencies = Currency.init();
    Scanner scanner = new Scanner(System.in);

    System.out.println("=====");
    System.out.println("      CURRENCY CONVERSION SYSTEM      ");
    System.out.println("=====|n");

    while (true) {
        System.out.println("== Currency Converter Menu ==");
        System.out.println("1. List all currencies");
        System.out.println("2. Convert currency");
        System.out.println("3. Exit");
        System.out.print("Choose an option: ");

        int choice = scanner.nextInt();

        switch (choice) {
            case 1:
                System.out.println("\nAvailable Currencies:");
                for (int i = 0; i < currencies.size(); i++) {
                    System.out.println((i + 1) + ". " + currencies.get(i).getName() +
}

```

```

    " (" + currencies.get(i).getShortName() + ")");
}

break;

case 2:

System.out.println("\nSelect source currency:");
for (int i = 0; i < currencies.size(); i++) {
    System.out.println((i + 1) + ". " + currencies.get(i).getName());
}

int srcIndex = scanner.nextInt() - 1;

System.out.println("Select target currency:");
for (int i = 0; i < currencies.size(); i++) {
    System.out.println((i + 1) + ". " + currencies.get(i).getName());
}

int tgtIndex = scanner.nextInt() - 1;

String sourceShortName = currencies.get(srcIndex).getShortName();
String targetShortName = currencies.get(tgtIndex).getShortName();

// Dynamic prompt
System.out.print("Enter the " + sourceShortName + " value you want to change to " +
targetShortName + ": ");
Double amount = scanner.nextDouble();

Double exchangeRate = currencies.get(srcIndex).getExchangeValues().get(targetShortName);
Double converted = Currency.convert(amount, exchangeRate);

System.out.println(amount + " " + sourceShortName +

```

```
" = " + converted + " " + targetShortName);
```

```
System.out.println("\nPress Enter to continue...");
```

```
try {
```

```
    System.in.read();
```

```
} catch (Exception e) {
```

```
    e.printStackTrace();
```

```
}
```

```
break;
```

*case 3:*

```
System.out.println("\n=====");
```

```
System.out.println(" Thank you for using Currency Conversion System ");
```

```
System.out.println("=====");
```

```
scanner.close();
```

```
return;
```

*default:*

```
System.out.println("Invalid choice. Try again.");
```

```
}
```

```
}
```

```
}
```

```

1 import java.util.ArrayList;
2 import java.util.HashMap;
3 import java.util.Scanner;
4
5 public class Currency {
6
7     private String name;
8     private String shortName;
9     private HashMap<String, Double> exchangeValues = new HashMap<String, Double>();
10
11    public Currency(String nameValue, String shortNameValue) {
12        this.name = nameValue;
13        this.shortName = shortNameValue;
14    }
15
16    public String getName() { return this.name; }
17    public String getShortName() { return this.shortName; }
18    public HashMap<String, Double> getExchangeValues() { return this.exchangeValues; }
19
20
21    public void defaultValues() {
22        switch (this.name) {
23            case "US Dollar":
24                this.exchangeValues.put("USD", 1.00);
25                this.exchangeValues.put("EUR", 0.94);
26                this.exchangeValues.put("GBP", 0.79);
27                this.exchangeValues.put("CHF", 0.90);
28                this.exchangeValues.put("CNY", 7.10);
29                this.exchangeValues.put("JPY", 150.00);
30                this.exchangeValues.put("INR", 83.50);
31                break;
32
33            case "Euro":
34                this.exchangeValues.put("USD", 1.06);
35                this.exchangeValues.put("EUR", 1.00);
36                this.exchangeValues.put("GBP", 0.84);
37                this.exchangeValues.put("CHF", 0.96);
38                this.exchangeValues.put("CNY", 7.55);
39                this.exchangeValues.put("JPY", 159.00);
40                this.exchangeValues.put("INR", 88.80);
41                break;
42
43            case "British Pound":
44                this.exchangeValues.put("USD", 1.27);
45                this.exchangeValues.put("EUR", 1.19);
46                this.exchangeValues.put("GBP", 1.00);
47                this.exchangeValues.put("CHF", 1.14);
48                this.exchangeValues.put("CNY", 9.00);
49                this.exchangeValues.put("JPY", 189.00);
50                this.exchangeValues.put("INR", 106.00);
51                break;
52
53            case "Swiss Franc":
54                this.exchangeValues.put("USD", 1.11);
55                this.exchangeValues.put("EUR", 1.04);
56                this.exchangeValues.put("GBP", 0.88);
57                this.exchangeValues.put("CHF", 1.00);
58                this.exchangeValues.put("CNY", 7.90);
59                this.exchangeValues.put("JPY", 165.00);
60                this.exchangeValues.put("INR", 92.50);
61                break;
62
63            case "Chinese Yuan Renminbi":
64                this.exchangeValues.put("USD", 0.14);
65                this.exchangeValues.put("EUR", 0.13);
66                this.exchangeValues.put("GBP", 0.11);
67                this.exchangeValues.put("CHF", 0.13);
68                this.exchangeValues.put("CNY", 1.00);
69                this.exchangeValues.put("JPY", 21.00);

```

```

70         this.exchangeValues.put("INR", 11.75);
71         break;
72
73     case "Japanese Yen":
74         this.exchangeValues.put("USD", 0.0067);
75         this.exchangeValues.put("EUR", 0.0063);
76         this.exchangeValues.put("GBP", 0.0053);
77         this.exchangeValues.put("CHF", 0.0061);
78         this.exchangeValues.put("CNY", 0.048);
79         this.exchangeValues.put("JPY", 1.00);
80         this.exchangeValues.put("INR", 0.56);
81         break;
82
83     case "Indian Rupee":
84         this.exchangeValues.put("USD", 0.012);
85         this.exchangeValues.put("EUR", 0.011);
86         this.exchangeValues.put("GBP", 0.0094);
87         this.exchangeValues.put("CHF", 0.011);
88         this.exchangeValues.put("CNY", 0.085);
89         this.exchangeValues.put("JPY", 1.80);
90         this.exchangeValues.put("INR", 1.00);
91         break;
92     }
93 }
94
95
96 public static ArrayList<Currency> init() {
97     ArrayList<Currency> currencies = new ArrayList<Currency>();
98
99     currencies.add(new Currency("US Dollar", "USD"));
100    currencies.add(new Currency("Euro", "EUR"));
101    currencies.add(new Currency("British Pound", "GBP"));
102    currencies.add(new Currency("Swiss Franc", "CHF"));
103    currencies.add(new Currency("Chinese Yuan Renminbi", "CNY"));
104    currencies.add(new Currency("Japanese Yen", "JPY"));
105    currencies.add(new Currency("Indian Rupee", "INR"));
106
107    for (Currency c : currencies) {
108        c.defaultValue();
109    }
110    return currencies;
111 }
112
113 public static Double convert(Double amount, Double exchangeValue) {
114     Double price = amount * exchangeValue;
115     price = Math.round(price * 100d) / 100d;
116     return price;
117 }
118
119
120 public static void main(String[] args) {
121     ArrayList<Currency> currencies = Currency.init();
122     Scanner scanner = new Scanner(System.in);
123
124
125     System.out.println("===== CURRENCY CONVERSION SYSTEM =====");
126     System.out.println("===== =====");
127     System.out.println("===== =====\n");
128
129     while (true) {
130         System.out.println("==> Currency Converter Menu ==>");
131         System.out.println("1. List all currencies");
132         System.out.println("2. Convert currency");
133         System.out.println("3. Exit");
134         System.out.print("Choose an option: ");
135
136         int choice = scanner.nextInt();
137
138         switch (choice) {

```

```

139     case 1:
140         System.out.println("\nAvailable Currencies:");
141         for (int i = 0; i < currencies.size(); i++) {
142             System.out.println((i + 1) + ". " + currencies.get(i).getName() +
143                                 " (" + currencies.get(i).getShortName() + ")");
144         }
145         break;
146
147     case 2:
148         System.out.println("\nSelect source currency:");
149         for (int i = 0; i < currencies.size(); i++) {
150             System.out.println((i + 1) + ". " + currencies.get(i).getName());
151         }
152         int srcIndex = scanner.nextInt() - 1;
153
154         System.out.println("Select target currency:");
155         for (int i = 0; i < currencies.size(); i++) {
156             System.out.println((i + 1) + ". " + currencies.get(i).getName());
157         }
158         int tgtIndex = scanner.nextInt() - 1;
159
160         String sourceShortName = currencies.get(srcIndex).getShortName();
161         String targetShortName = currencies.get(tgtIndex).getShortName();
162
163         System.out.print("Enter the " + sourceShortName + " value you want
164             to change to " + targetShortName + ": ");
165         Double amount = scanner.nextDouble();
166
167         Double exchangeRate = currencies.get(srcIndex).getExchangeValues().get(targetShortName);
168         Double converted = Currency.convert(amount, exchangeRate);
169
170         System.out.println(amount + " " + sourceShortName +
171                             " = " + converted + " " + targetShortName);
172
173         System.out.println("\nPress Enter to continue...");
174         try {
175             System.in.read();
176         } catch (Exception e) {
177             e.printStackTrace();
178         }
179         break;
180
181     case 3:
182         System.out.println("\n=====");
183         System.out.println(" Thank you for using Currency Conversion System "
184             );
185         System.out.println("=====");
186         scanner.close();
187         return;
188
189     default:
190         System.out.println("Invalid choice. Try again.");
191     }
192 }
193 }
```

# OUTPUT

C:\Users\Rahul\Desktop\JAVA>javac Currency.java

C:\Users\Rahul\Desktop\JAVA>java Currency

=====

CURRENCY CONVERSION SYSTEM

=====

==== Currency Converter Menu ====

- 1. List all currencies
- 2. Convert currency
- 3. Exit

Choose an option: 2

Select source currency:

- 1. US Dollar
- 2. Euro
- 3. British Pound
- 4. Swiss Franc
- 5. Chinese Yuan Renminbi
- 6. Japanese Yen
- 7. Indian Rupee

1

Select target currency:

- 1. US Dollar
- 2. Euro
- 3. British Pound
- 4. Swiss Franc
- 5. Chinese Yuan Renminbi
- 6. Japanese Yen
- 7. Indian Rupee

7

Enter the USD value you want to change to INR: 56

56.0 USD = 4676.0 INR

Press Enter to continue...

==== Currency Converter Menu ====

- 1. List all currencies
- 2. Convert currency
- 3. Exit

Choose an option: 3

=====

Thank you for using Currency Conversion System

=====

```
C:\Windows\System32\cmd.e + ^ C:\Users\Rahul\Desktop\JAVA>javac Currency.java
C:\Users\Rahul\Desktop\JAVA>java Currency
=====
      CURRENCY CONVERSION SYSTEM
=====

==== Currency Converter Menu ===
1. List all currencies
2. Convert currency
3. Exit
Choose an option: 2

Select source currency:
1. US Dollar
2. Euro
3. British Pound
4. Swiss Franc
5. Chinese Yuan Renminbi
6. Japanese Yen
7. Indian Rupee
1
Select target currency:
1. US Dollar
2. Euro
3. British Pound
4. Swiss Franc
5. Chinese Yuan Renminbi
6. Japanese Yen
7. Indian Rupee
7
Enter the USD value you want to change to INR: 56
56.0 USD = 4676.0 INR

Press Enter to continue...

==== Currency Converter Menu ===
1. List all currencies
2. Convert currency
3. Exit
Choose an option: 3

=====
      Thank you for using Currency Conversion System
=====
```